Never Use Unvalidated Input as Part of a Directive to any Internal Component

William L. Fithen, Software Engineering Institute [vita[3]]

2005-10-03

Using unvalidated input as part of a directive or command to a subsystem can introduce vulnerability.

Description

"Injection" is a special kind of input validation vulnerability. It is special in that the input that is not validated is later used as input to some embedded system, such as a script processor or a database.

Warning

The several pieces of advice presented here that relate to "injection" are very ambiguously related. Their coverage in literature tends to define them somewhat arbitrarily by either their engineering cause of the pattern of attack carried out by an adversary to exploit them. These two ways of describing vulnerabilities lead to quite different coverage and to different categorizations. Even the term "injection"—which is fairly uniformly used—is strange to engineers since the use of the term implies a clearly adversarial perspective, rather than an engineering one.

Command Injection

When a program accepts unvalidated input from a user and uses that input to form a dynamic command to be executed by a "command-oriented" subsystem, a vulnerability may exist.

The usual context for this class of vulnerability is a program that invokes a command processor (e.g., UNIX shell) with commands derived from user input [[VU#502328[11]], VU#195371[12]].

SQL Injection

When a program accepts unvalidated input from a user and uses that input to construct a dynamic SQL query to an SQL database, a vulnerability may exist.

The usual context for this class of vulnerability is a web server that accepts input from browsers, uses that input to construct queries against a server-side database, and formats the query response to the browser. The browser user may construct crafted input that, when embedded in a string that is interpreted as an SQL query, performs database operations that are not intended.

Cross-Site Scripting

The security community that has not completely settled on the definition of cross-site scripting. In its

---

3. daisy:320 (Fithen, William L.)

11. #refs

12. #refs

Never Use Unvalidated Input as Part of a Directive to any Internal Component
ID: 342 | Versie: 4 | Datum: 4/04/06 14:34:41

1

simplest form,

Cross-site scripting is possible when

1. An adversary tricks a victim into clicking on a link that he crafted and presented to the victim (via a web server or email),

2. The link contains a URL with embedded malicious script (typically as a query string), and

3. The URL refers to host that echoes input back to a browser without input validation.

When the victim clicks the link, he is referred to the host in the URL, the host processes the query string and echoes it to the victim's browser, and the victim's browser executes the malicious script.

This is an unusual vulnerability because the system at fault, the web site not doing input validation, is not the victim of attack.

The only remedy for this vulnerability is for the web site to perform adequate input validation.

References

[Anley 02]	Anley, Chris. *Advanced SQL Injection In SQL Server Applications*. http://www.nextgenss.com/papers/advanced_sql_injection.pdf (2002).
[Apache 00]	apache.org. *Cross Site Scripting Info*. http://httpd.apache.org/info/css-security/ (2000).
[CA-2000-02]	cert.org. *CERT® Advisory CA-2000-02 Malicious HTML Tags Embedded in Client Web Requests*. http://www.cert.org/advisories/CA-2000-02.html (2000).
[CERT 00]	cert.org. *Understanding Malicious Content Mitigation for Web Developers*. http://www.cert.org/tech_tips/malicious_code_mitigation.html (2000).
[Howard 02]	Howard, Michael & LeBlanc, David. *Writing Secure Code*, 2nd. edition. Redmond, WA: Microsoft Press, 2002.
[Integrigy 04]	integrigy.com. *An Introduction to SQL Injection Attacks for Oracle Developers*. http://www.integrigy.com/info/IntegrigyRethinkingSQLInjectionAttacks.pdf (2004).
[VU#195371]	Havrilla, Jeffrey S. *Vulnerability Note VU#195371: SGI IRIX rpc.xfsmd does not filter shell metacharacters from user input before invoking popen() function*. http://www.kb.cert.org/vuls/id/195371 (2003).
[VU#502328]	MacInnis, Ken. *Vulnerability Note VU#502328: SquirrelMail vulnerable to command injection because of flawed input checking in S/MIME plug-in*. http://www.kb.cert.org/vuls/id/502328 (2005).

SEI Copyright

Velden

Naam	Waarde
Copyright Holder	SEI

Velden

Naam	Waarde
is-content-area-overview	false
Content Areas	Knowledge/Guidelines
SDLC Relevance	Implementation
Workflow State	Publishable

---

1. http://www.sei.cmu.edu/about/legal-permissions.html